*Applied Mathematics & Information Sciences* 5 (2) (2011), 179S-185S
– An International Journal

# Research on Implement Snapshot of pNFS Distributed File System

Liu-Chao, Zhang-Jing Wang, Liu Zhenjun , Wang-Chao, Zhang-Jun Wei
Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
*LiuChao@nrchpc.ac.cn*

Received Jun 8, 2010; Revised Jan 20, 2011

**Abstract:** Snapshot plays an increasingly important role in the online-backup and data-protection of distribute file systems. As a new generation standard of NFS, pNFS doesn't provide design scheme on snapshot concretely. And the current snapshot implementations depending on their own file system architecture heavily, can't be used in pNFS. This paper takes a careful study on pNFS structure , adopts a suitable metadata organized solutions, proposes an method to get a consistent state of file system based on COW and a novel source-reserved read/write process flow, which resolve the problem of interaction between client and server, consistence and efficient of creating snapshots. To our knowledge, we firstly implement snapshot function of pNFS in Linux kernel 2.6 and the tests which shows creating snapshot has little affect on performance of pNFS. Snapshot is quick to create(less than 1 second) and can be created frequently (one snapshot per second) without serious performance degradation(less than 10% degradation to write bandwidth).

**Keywords:** snapshot, pNFS, Distributed file system (DFS), copy on write(COW)

## 1 Introduction

Snapshot defines that a fully usable copy of a defined collection of data that contains an image of the data as it appeared at a single instant in time [1]. Differing from the current local file system or virtual block device snapshot, the distribute file system now facing the following problems and challenges: Firstly ,in the distributed file system with the out-of-band accessing mode, client can read or write storage devices directly. Secondly, because of computer failure or network interrupt and other abnormalities, it is awfully difficult to get the consistent state of system. Thirdly, because that distributed file system needs to provide continuous service, ensuring the consistency of snapshot becomes a serious problem.

pNFS (parallel network file system NFSv4.1)[2]distributed file system framework, is an extension of NFSv4 . As the future of NFS, it inherits the advantage which pNFS will be accepted by most of users as a standard. In the near future, Linux kernel may make it to be a necessary part, every file system should support it. Taking into consideration that creating snapshot should be compatible with the pNFS standard, it brings a new challenge to us.

In the context of our knowledge , we highlight three main contributions in this paper, firstly, we implement the high performance pNFS distributed file system with snapshot function under block layout protocol for the first time，secondly，we propose a method based on all data COW to get the consistence state, Finally, we put forward a novel resource-reserved read/write process flow for the distributed snapshot system. Our tests shows creating snapshot doesn't affect the performance of

pNFS distribute file system. The rest of the paper is organized as follows. In the next section，we provide a brief background of pNFS, and the related work about the snapshot in other distributed file systems. Section 3 explains in detail about the key technologies to create snapshots in pNFS. Section 4 presents some experimental results, Finally, section 5 presents our conclusions.

## 2 Backgrounds and Related work

The classical NFS has been reviewed consistently. However, its data throughput and scalability cannot satisfy the growing demands of larger clusters because the data and metadata has to pass through a single NFS server. Under such circumstance, some research institutions and incorporations advocate pNFS, which is based on NFSv4 and out-band data transmission, allowing the clients to access the storage device directly. It profoundly reduces the possibility of the server becoming a bottleneck of the system and provides much higher scalability and throughput. Meanwhile, With the expectation to be the standards of distributed file system in future, pNFS has win much support from IBM, NetApp, panasas, EMC etc. pNFS can provide higher availability and reliability with snapshot functionality.

### 2.1 pNFS

pNFS consists of three main components: server, client and storage-node. Server manages the metadata, client issuing I/O requests and storage-node responsible for storing data. The whole procedure is like this: a client obtains a layout from the metadata server, than this client will read or write the storage-node according the layout information.

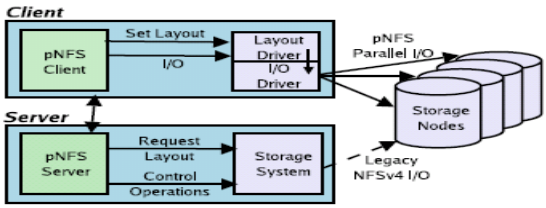


Figure 2.1 pNFS Arichitecture

The implementation of snapshot relies on the mechanism of pNFS which are all about its layout. The layout of pNFS defines how the blocks of a specific file distribute on a single or multiple storage devices. It contains the following parts.

1. Layout Type. This represents the type of storage devices that the clients can access. This paper will focus on block device on which to create snapshots.

2. Layout IOmode. This defines the layout permissions. Table 2.1 shows that four kinds of permissions in the block layout type protocol.

Table 2.1 Layout explanations

| Read/write permission | Explanation |
|---|---|
| READ_WRITE_DATA | Range indicated by layout can be read or written |
| READ_DATA | Range indicated by layout can only be read |
| INVALID_DATA | Represents the new allocated but uninitialized data |
| NONE_DATA | Represents the file hole |

3. The logical offset and range in a file. This part demonstrates the logical area of a file corresponding to the layout information.
4. The layout content. It describes the physical device distribution of a file system.

pNFS contains four operations for interaction :

1. Obtaining a Layout. The client gets the layout of a file through LAYOUTGET.
2. Commit a layout. The client synchronizes the layout with the server by the operation LAYOUTCOMMIT after it has changed it.
3. Return a layout. The client returns the layout previously obtained to the server by the operation LAYOUTRETURN.
4. Recall a layout. The server revokes the layout previously granted to the client by the operation CB_LAYOUTRECALL.

### 2.2 Related Work

Lots of work has been done in the snapshot of DFSs. Google FS's snapshot [3] is also based on the technology of Copy on Write (COW). General Parallel File System (GPFS) [4] can snapshot the whole file system and roll back files to any snapshots. Lustre [5], developed by Cluster file system, uses Object based Storage Devices (OSD). The mapping from file to the block position is done by the object based storage devices, which support snapshot. Write Anywhere File Layout (WAFL) is a file system used by NAS application of NetApp, using Checkpoint and COW in its snapshot. But these snapshots are highly coupling with their systems and cannot be well integrated into pNFS framework.

## 3 Key Technologies on Snapshot

### 3.1 Metadata Organization

The organization and management of metadata is of vital magnitude in file system. In DFSs, to prevent the concurrency and synchronization complexities of distributed metadata updates, centralized metadata management is preferred. As a flexible distributed file system framework, pNFS's block layout protocol supports manifold back-end file systems. These file systems, which correspond to the metadata server in pNFS architecture, are responsible for the organization and representation of metadata. It also has the main functionalities for creating snapshots.

Due to its highly efficient storage and fast retrieval, B+ tree is widely used in metadata organization. Rodeh'paper provides with us an improved B+ tree structure, which ensures good performance under COW, and further, when combined with reference count mechanism, a well-establish snapshot functionality [6]. This advancement relies on the following skills:

1. top-down updates.
2. Leaf links removed.
3. Lazy updates of reference count.

The question that how to create snapshots on B+ trees and the details of these skills are beyond the scope of this paper. Readers of interest can refer to the Rodeh' paper [6].

### 3.2 Resource-reserved Read/Write Process

In distributed snapshot file system, the read/write process is essential to keep snapshot semantics. Client mounts the file system exported by pNFS,， As Figure 3.1 shows:


Figure 3.1 pNFS read/write process

1. Client mount the file system and get the device lists and device information.

If client want to read a file:

2. Client get the layout, server receives the request and inquires mapping information, if demands, reserving the disk space information of the file in memory, then sends the reply to client.
3. Client resolve the layout information, find the physical location in the devices.
4. Client read the data from the devices

If client want to write a file, continuing the 3th step, then:

5. Client write the data to the devices directly
6. Client commit the modifications to the Metadata server.

### 3.3 Get Consistence State

The obvious difference between the block device snapshot and file system snapshot is that the file system should ensure the consistency when creating snapshot. Because of the running file system always updating its data, if when these operations doesn't finish, the snapshot will contain these unfinished data, and data set of snapshot is inconsistent. We work out a very simple approach based on all data COW to get a consistent state, taking a look at table 3.1 about layout type.

Table 3.1 Layout types of interaction

| Request the layout type of file | Layout type returned |
|---|---|
| Read/ Write Hole | NONE_DATA INVALID_DATA |
| Read/Write Common layout | READ_DATA/READ_WRITE_DATA |
| Read/Write Snapshot layout | READ_DATA/READ_DATA&INVALID_DATA |

Client can modify the corresponding part according to READ_WRITE_DATA layout . So when creating a snapshot , it must recall this layout Recalling a layout is long time process in DFS. Taking considerations to performance, our approach to get the consistent state is that, with all data cow, avoiding granting this layout to client.

### 3.4 Create Snapshots

If do the COW to the all data at any time, we can create snapshots in Meta server directly, but the problem of file fragmentation become serious. If segmentation isn't referred by a snapshot, server can grant the READ_WRITE_DATA to client, and the client update data in the same place. Under this circumstance，the process to create snapshots shows as follows, First, Server enables all data cow and recall the layout asynchronously, then creates snapshot, at last, disables function of all data cow.

Because that the modifications to metadata and creation of snapshots are protected by the transaction mechanism of metadata server, the file mapping relation of active file system and snapshot can keep consistence.

## 4 Implementation and Experimental Evaluation

We used the open-source pNFS as DFS framework and btrfs as the back-end metadata file system. With the Linux core 2.6.32, we exported the pNFS module from btrfs, Our metadata server uses 2.2GHz Xeon E5410 CPU and 4G memory. Two machines of the same configuration are used as clients. Three shared disks (750G, 7200RPM, STATII) are configured as RAID 0, exported by the fiber for MDS and the clients. We tested the performance of metadata server and the disk data I/O.

### 4.1 Metadata performance test

We want to show by this test that our implementation of snapshot in pNFS doesn't exert extra influence on the creation speed of files and directories. The test recursively builds a directory tree. In each of the directory, a certain amount of files of designated size will be created and we just count how much time these operations will take eventually. Shows in Figure4.1, we can see that the number of snapshots doesn't compromise, if not improve, the performance of the metadata.

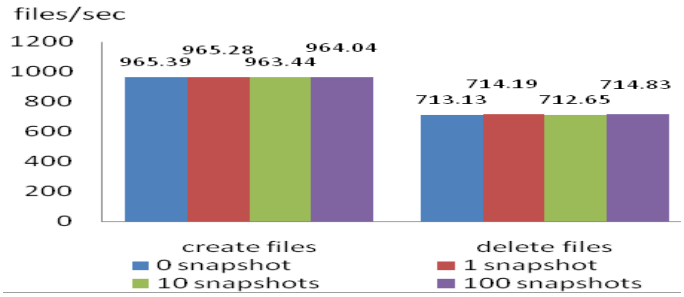


Figure 4.1 The speed of create and delete files under different number of snapshots

### 4.2 Data performance test

The data performance is tested under different number of snapshots and different frequency of creating snapshots. Tests shows the performance of reading or writing a file of 8G size with 1M block size at a time. Figure 4.2 shows that there are few varies among these tests. It indicates that the frequency of creating snapshot has no much negative effects on bandwidth performance.

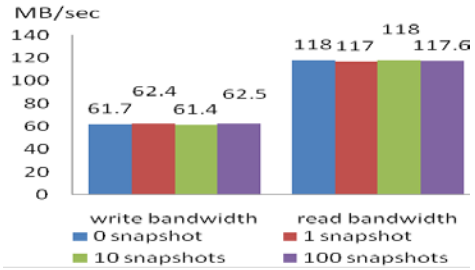


Figure 4.2 Read/ Write bandwidth

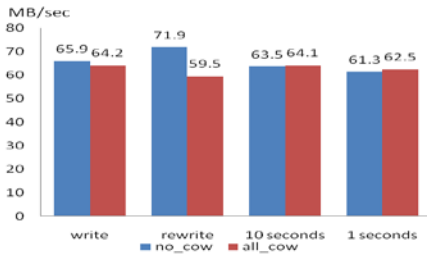


Figure 4.3 no_cow and all_cow write bandwidth

Without the COW for any data at any time (no_cow), we tested the initial write performance before creating any snapshots, the rewrite performance, the write performance when creating a snapshot every 1 or 10 seconds. With the COW for all data (all_cow), we made the similar tests. As Figure4.3 shows, frequent creation of snapshots also play a minor role in the writing performance.

## 5 Conclusion

In this paper, we study on the key technologies on implementing snapshot under pNFS framework, and the snapshot shows very good performance. we highlight the contributions in this paper, firstly, we implement pNFS distributed file system with snapshot function under block layout protocol for the first time，secondly，we propose a method based on all data COW to get the consistence state, Finally, we put forward a novel source-reserved read/write process flow for the snapshot system. The test results indicate that research methods are effective under the block layout protocol of pNFS.

## Acknowledgements

This work is supported in part by the National High-Tech Research and Development Plan of China under grants No. 2009AA01A403, China Information Security Special (NDRC) fund for Disaster Backup and Recovery Standard Architecture Construction.

## References

[1] SNIA. http://www.snia.org/education/dictionary/s/. 2010
[2] S. Shepler, M. Eisler, D. Noveck, Network File System (NFS) Version 4 Minor Version 1 Protocol.http://www.ietf.org/rfc/rfc5661.txt, January 2010.
[3] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, The Google File System , in: Symposium on Operating Systems Principles (SOSP'03).pp. 29-43. 2003.
[4] Frank Schmuck and Roger Haskin, GPFS: A Shared-Disk File System for Large Computing Clusters, in: Proceedings of the Conference on File and Storage Technologies, pp. 231–244.January 2002.
[5] Cluster File Systems Inc., Lustre: A Scalable, High-Performance FileSystem, www.lustre.org, 2002.
[6] Ohad Rodeh, B-trees, Shadowing, and Clones, In: ACM Transactions on Computational Logic, Vol. V, No. N, August 2007.

LiuChao received the bachelor degree in information security from HuNan University (HNU), China in 2009, He is currently a Master Student in the ICT/CAS. His research interests include the areas of computer architecture, distributed storage systems and database system.